Topic:	Processing Touch Input and UART Communication
Author:	JOKI
Date:	05.07.2021

0. Before you start

This document will give you an overview of the source code for the example package Drawing Text and Images. Before you can work with it you need to set up your working environment as explained in the document “ExamplePackage_GettingStarted”. Make sure you have read this document beforehand and executed all the steps to configure your Workbench. Especially, that you have added all the paths to the *Source Location* and *Includes* properties.

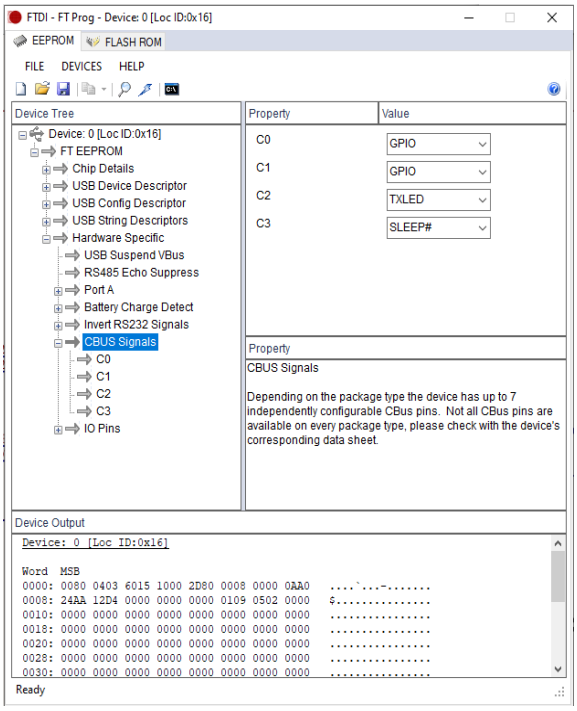
For a more detailed explanation of how the microcontroller works, please refer to the STM32F429 Reference Manual (RM0090) and the Standard Peripheral Library documentation provided by STMicroelectronics.

1. Introduction

In this example package we will work with the touch panel of our display module, which is mounted on top of the LCD-module. The touch panel has its own control board which is connected to the microcontroller via I²C. When you touch the display, you can get the position of this touch event. The same applies to a release event. This information, the position and the type of event, is then sent over UART to the on-board FTDI-chip, which converts the data input to the USB-format and sends the data through the USB-Type-B socket to a connected PC.

2. Additional Required Software

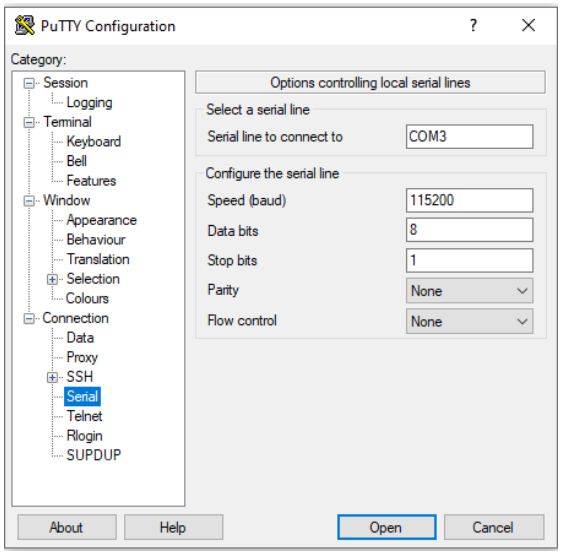
2.1 FT_PROG

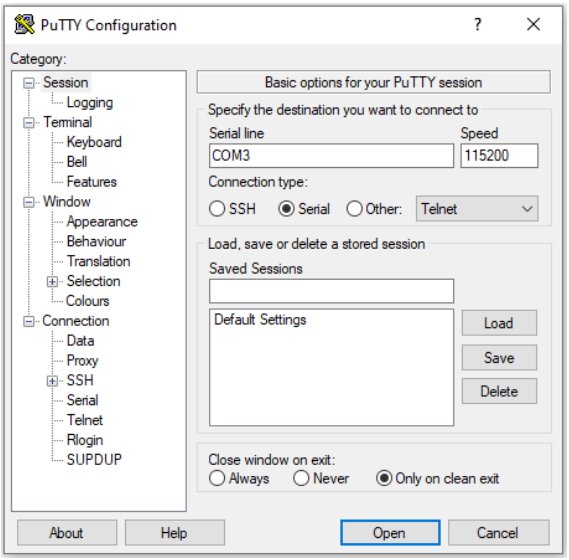


You may or may not need to configure the FTDI-chip. To make sure, that the chip is configured correctly you should download the free EEPROM programming utility FT_PROG by FTDI (https://ftdichip.com/utilities/). Connect your computer with a USB-Type-A to USB-Type-B cable to the display board and make sure that the board is connected to its power supply. Start FT_PROG and click on the button with the magnifying glass. This will start the scan process for connected devices. Then select *Hardware Specific -> CBUS Signals*. Now check if values for the properties C0 and C1 on the right panel are set to *GPIO*. If not, change this. Afterwards click on the lightning button and click *Program* to start the programming process. After the programming has ended successfully, turn the power off, wait a few seconds and turn it on again. The FTDI-chip should now be configured properly.

2.2 PuTTY

To read the data, that is transmitted by the microcontroller to the PC, you need a program to establish the required serial connection. We recommend the free software PuTTY for this. When you opened the program, you first need to configure the serial connection. Click on the tab *Serial* and configure the serial line with the values you see in the left image below. They should match with the values we used to initialize the UART interface of the microcontroller (see */src/driver/d_uart.c -> void uart_init(void)*). Next, click on the tab *Session* and choose the connection type *Serial*. The other parameters will change automatically. Now, you can press *Open* to start the connection. A terminal window should pop up.





3. Explanation of Example Code

3.1 Main function

main.c

```
/*****************************************************************
 *
 * Function: int main(void)
 *
 * Summary: Entry point of display control application
 *
 * Parameters:
 *
 * Return:
 *
 * Revision History:
 * Creation Date: 08042019
 *
 *****************************************************************/
int main(void) {
    /* Initialize the peripherals */
    HW_Init();
    Touch_FT5316_init();
    display_init();

    DMA2D_Fill_Color(0xFFFFFFFF, Layer_1, Buffer_1);
    DMA2D_write_string("Touch Me!", (HDP-9*16)/2, VDP/2-11, 0xFF000000, &courier_new, Layer_1, Buffer_1, Layer_1, Buffer_1);
    for(;;){
        process_touch_input();
    }
    return 0;
}
```

We will start our walk through the code at the main function since this gives a perfect overview of the steps that we will take. At the beginning, the used peripherals are initialized. *HW_Init()* initializes and turns on the display backlight. Also, it sets up the different system clocks and the timer, which counts every microsecond, and the UART connection.

Then we initialize the touch panel and the display which also includes the initialization of the SDRAM, because this is where the buffer for the displayed images will be located. The DMA2D is used to send the image data directly from the buffer to the LCD, which speeds up the image manipulation. Part of the display initialization is the drawing of the EBS-SYSTART logo and the logo of our 2in1 display line.

After the initialization process, we start with drawing the background that asks you to touch the display. The following main loop is the frame for the main task of the program, which is to react to touch inputs.

3.2 Function: process_touch_input

```
/*************************************************************
 *
 * Function:  void process_touch_input(void)
 *
 * Summary:   Read touch input and populate the touch response struct
 * Parameters:
 * Return:
 *
 * Revision History:
 * Creation Date: 08042019
 *
 *************************************************************/
void process_touch_input(void) {
    static bool state = RELEASE;
    TouchXY_t txy;
    static int touch_snd = 0;
    static tTimer timeout;

    if (touch_snd) {
        if(timer_get_time_ms() - timeout > 500)
            return;
        touch_snd = 0;
    }

    txy = Touch_FT5316_getTouchXY();
    // if touch event
    if (txy.event == TOUCH) {
        if (state != TOUCH) {
            state = TOUCH;
            send_touch_response(&txy);
            touch_snd = 1;
            //Area check
            if(AreaCheck(0, 0, HDP/2, VDP/2)){
                DMA2D_Fill_Color(0xFF00FF00, Layer_1, Buffer_1);
            }
            else{
                DMA2D_Fill_Color(0xFFFF0000, Layer_1, Buffer_1);
            }
            timeout = timer_get_time_ms();
        }
    }
    //else if release event
    else if (txy.event == RELEASE) {
        if (state != RELEASE) {
            state = RELEASE;
            send_touch_response(&txy);
            touch_snd = 1;
            timeout = timer_get_time_ms();
            DMA2D_Fill_Color(0xFFFFFFFF, Layer_1, Buffer_1);
            DMA2D_write_string("Touch Me!", (HDP-9*16)/2, VDP/2-11, 0xFF000000, &courier_new, Layer_1, Buffer_1, Layer_1, Buffer_1);
        }
    }
    //else if timeout error in i2c communication
    else if (txy.event == 0xFF) {
        send_touch_response(&txy);
        timeout = timer_get_time_ms();
    }
}
```

This function fulfills the main task of this example code. This function will be called repeatedly and every time it asks if there was a touch/release event. If there was one, it compares the event type with the previous and only when they differ, it will store the information in the *touch_resp_t* struct and send it via UART. Therefore, only single-finger-input is supported.

Every time you touch the display, an area check will be executed. This means that the coordinates of the last touch input will be compared with the parameters of the function *AreaCheck()*. If the touch input is inside these limits, the display will turn green otherwise it will turn red. When you release the display, the previous text will be displayed again.

3.3 Function: send_touch_response

```
/*************************************************************
 *
 * Function: comm_err_t send_touch_response(touch_resp_t *touch_resp)
 *
 * Summary: Pack the touch coordinates into uart command format and
 *          send the command over uart
 *
 * Parameters:
 *     touch_rep - pointer to the struct holding touch coordinates
 *                 and event type
 *
 * Return:
 *  0 - if success
 *
 * Revision History:
 * Creation Date: 08042019
 *
 *************************************************************/
comm_err_t send_touch_response(touch_resp_t *touch_resp) {
    char output[49];
    tTimer send_timeout;

    if(touch_resp->event_type == 0x01){
        sprintf(output, "X,Y-Coordinates: (%3i,%3i) EVENT-TYPE:   TOUCH\r\n", touch_resp->xpos, touch_resp->ypos);
    }
    else if (touch_resp->event_type == 0x00){
        sprintf(output, "X,Y-Coordinates: (%3i,%3i) EVENT-TYPE: RELEASE\r\n", touch_resp->xpos, touch_resp->ypos);
    }
    else if (touch_resp->event_type == 0xFF){
        sprintf(output, "X,Y-Coordinates: (---,---) EVENT-TYPE:   ERROR\r\n");
    }


    send_timeout = timer_get_time_ms();
    uart_SendString(COM1, output, 0);
    if(timer_get_time_ms() - send_timeout > 10) return FCN_TIMEOUT;


    return FCN_OK;
}
```

This function packs the information about the touch/release event, which is stored in the *touch_resp_t* struct, into a string. This string is then sent via UART to the PC.

Note, that the connection name *COM1* doesn't need to be the same as the one you may see on your computer. Here, *COM1* only specifies which UART pins of the microcontroller are used. The FTDI-chip for the USB-communication is connected to UART1, therefore the connection name is *COM1*.

4. Ideas for Exercise Project

Here, we want to give you some suggestions how you could modify our example code and make your own little project.

In this example package we used the UART connection to send the touch/release inputs to a PC. Now you could try the other direction and send data via UART to the microcontroller and process it, e.g. show the text you type on your keyboard on the display.

You could also modify the other example packages "Digital Picture Frame" and "Drawing Text and Images" and include touch sensitivity. You can find suggestions in the according documentation.